Abstract

Software reliability is one of the most important software quality attributes. It is an external quality attribute, which relates internally to the notion of program faults or defects. In order to assess software reliability, one of methods is to apply Software Reliability Growth Models (SGRM). More than 200 SGRM exist. However these approaches are black-box based i.e., the software system is considered as a whole and only its interactions with the outside world are modeled, without looking into its internal structure. These are mostly employed at the later stage of the software development life cycle, before which many critical design decisions are made. Identification of significant problems during implementation or operation can lead to re-engineering of large parts of the system, which has been shown to be prohibitively costly. Further they are based on unrealistic assumptions. Therefore, several recent approaches have begun to quantify software reliability at the level of architectural models, or at least in terms of high-level system structure. Some of the existing approaches are based on Markov Model. It has been experienced that mostly software fails because of ambiguity in the requirements or defect in the design. Ambiguous requirements may also lead to the design defect. So, a model should be explainable to all the stakeholders to understand all the functional requirements. Further it will give an assurance to the client that all the functional requirements of software have been taken care of. UML has proved to be a general-purpose modeling language in the field of software engineering that can be understood by all the stakeholders. UML has an ability to model the scenarios of the system.

The possibility to extend the UML to convert into a probabilistic reliability model is shown. The converted reliability model is a Markov Model, which can be easily plugged into the existing approaches of reliability prediction based on Markov Model.

The limitations of the Markov reliability Model for early prediction are also concluded. They are incapable to model the concurrency and typically limited to modeling the probability of changes in the system with the exponential distribution. Further, in

these models, the reliability is the function of transition probabilities in between the states of the Markov Chain, which are either assumed or computed based on the operational profile. Analytical approach of predicting transition probabilities does not give accurate prediction. On the other hand, operational profile can be collected in the last stage of the software development life cycle or during the software operational time. Hence transition probabilities, based on operational profile does not provide early estimate of reliability.

It is shown that these limitations can be addressed by modeling a system through Petri net. We use the tool TimeNET for steady state distribution of the transition rates in between the states of Markov model, through which the transition probabilities have been derived.

Additionally, the reliability requirements of each component of a software system may not be necessarily same. Also during the system operation, its dynamics changes, which may affect the reliability of the components and hence of the overall system. Therefore, there is a requirement to have a methodology for updating the reliabilities of each component and system as well for corrective action. This motivates us to propose an approach for reliability updation of the components. Bayesian approach is used to address these issues.

All the above proposed approaches are validated with the real time safety critical system of Indian Nuclear Power Plant along with some noticeable findings.